IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of: Sheueling Chang Shantz, et al.	§	Examiner: Johnson, Carlton
	§	
	§	Group Art Unit: 2136
Serial No. 10/626,420	§	
	§	Atty. Dkt. No.: 6000-32301
Filed: July 24, 2003	§	
	§	
For: Method and Apparatus for Implementing Processor Instructions for Accelerating Public-Key Cryptography	§	

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed December 20, 2007, Appellants present this Appeal Brief. Appellants respectfully request that the Board of Patent Appeals and Interferences consider this appeal.

I. REAL PARTY IN INTEREST

As evidenced by the assignment recorded at Reel/Frame 014320/0118, the subject application is owned by Sun Microsystems, Inc., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and now having its principal place of business at 4150 Network Circle, Santa Clara, CA 95054.

II. RELATED APPEALS AND INTERFERENCES

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-65 are pending in the application and stand finally rejected. The rejection of claims 1-65 is being appealed. A copy of claims 1-65 is included in the Claims Appendix herein below.

IV. STATUS OF AMENDMENTS

No amendments have been submitted subsequent to the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 is directed to a method implemented in a device supporting a cryptography application. (*See, e.g.,* title; p. 1, paragraph [1002]; p. 2, paragraph [1005]; p. 8, paragraphs [1057]-[1058]; and p. 16, paragraphs [1082]-[1083].)

The method includes, in response to executing a single arithmetic instruction, multiplying a first number by a second number, and adding implicitly a partial result from a previously executed single arithmetic instruction to generate a result that represents the first number multiplied by the second number summed with the partial result. (*See, e.g.,* p. 2, paragraph [1006]; FIG. 4A, which illustrates the *umulxc* instruction; and p. 12, paragraphs [1069]-[1070] and Table 1.)

The partial result comprises a high order portion of a result of the previously executed single arithmetic instruction. (*See, e.g.*, p. 12, Table 1, description of previous carry saved in *exc.*)

The method also includes, in response to executing the single arithmetic instruction, storing at least a portion of the generated result, and using the stored at least a portion of the generated result in a subsequent computation in the cryptography application. (*See, e.g.,* p. 2, paragraph [1006]; p. 12, paragraphs [1069]-[1070], multiply-chaining for multi-word computations; and p. 8, paragraph [1057], the application of multiply-chaining for accelerating public-key computations.)

Independent claim 18 is directed to a method implemented in a device supporting a cryptography application. (*See, e.g.,* title; p. 1, paragraph [1002]; p. 2, paragraph [1005]; p. 9, paragraphs [1057]-[1058]; and p. 16, paragraphs [1082]-[1083].)

The method includes, in response to executing a single arithmetic instruction, multiplying a first number by a second number, adding implicitly a partial result from a

previously executed single arithmetic instruction and a third number to generate a result that represents the first number multiplied by the second number summed with the partial result and the third number. (*See, e.g.*, p. 2, paragraph [1007]; FIGs. 4B and 4C, illustrating two different embodiments of the *umulxck* instruction; p. 13, paragraph [1073] – p. 14, paragraph [1076]; and p. 14, Table 2.)

The partial result comprises a high order portion of a result of the previously executed single arithmetic instruction. (*See, e.g.*, p. 14, Table 2, description of previous carry saved in *exc.*)

The method also includes, in response to executing the single arithmetic instruction, storing at least a portion of the generated result and using the stored at least a portion of the generated result in a subsequent computation in the cryptography application. (*See, e.g.,* pp. 2-3, paragraph [1007]; and p. 15, paragraph [1077], multiply-accumulate-chaining for multi-word multiplication, and its application to public-key computations.)

Independent claim 43 is directed to a processor that includes an arithmetic circuit. (*See, e.g.*, p. 3, paragraph [1008]; FIG. 15, multiply-and-accumulate circuit; and p. 29, paragraph [1124].)

The processor is configured to be responsive to execution of a single arithmetic instruction to cause the arithmetic circuit to multiply a first number and a second number and to add implicitly a high order portion of a partial result from a previously executed single arithmetic instruction, thereby generating a result that represents the first number multiplied by the second number summed with the high order portion of the partial result. (*See, e.g.*, p. 3, paragraph [1008]; FIG. 15, multiply-and-accumulate circuit; and p. 29, paragraph [1124].)

The processor is further configured to store at least a portion of the generated result and to use the stored at least a portion of the generated result in a subsequent computation. (*See, e.g.*, p. 3, paragraph [1008]; FIG. 15, multiply-and-accumulate circuit; p. 29, paragraph [1124], upper 64-bits of result output to *exc* register 1058; p. 12, paragraphs [1069]-[1070], multiply-chaining for multi-word computations; and p. 8, paragraph [1057], the application of multiply-chaining for accelerating public-key computations.)

Independent claim 50 is directed to a processor that includes an arithmetic circuit. (*See, e.g.*, p. 3, paragraph [1009]; FIG. 20, multiply-and-accumulate circuit; and pp. 30-31, paragraph [1130].)

The processor is configured to be responsive to execution of a single arithmetic instruction to cause the arithmetic circuit to multiply a first number and a second number, to add a third number, and to implicitly add a high order portion of a previous result from a previously executed single arithmetic instruction, thereby generating a result that represents the first number multiplied with the second number, summed with the high order portion of the previous result and with the third number. (*See, e.g.*, p. 3, paragraph [1009]; FIG. 20, multiply-and-accumulate circuit; and pp. 30-31, paragraph [1130].)

The processor is further configured to store at least a portion of the generated result and to use the stored at least a portion of the generated result in a subsequent computation. (*See, e.g.*, p. 3, paragraph [1009]; FIG. 20, multiply-and-accumulate circuit; pp. 30-31, paragraph [1130], upper 64-bits of result output to *exc* register 2005; and p. 15, paragraph [1077], multiply-accumulate-chaining for multi-word multiplication, and its application to public-key computations.)

Independent claim 57 is directed to a computer-readable storage medium comprising program instructions executable by a processor to implement a cryptography

application. (*See, e.g.,* title; p. 1, paragraph [1002]; p. 2, paragraph [1005]; p. 3, paragraph [1010]; p. 8, paragraphs [1057]-[1058]; and p. 16, paragraphs [1082]-[1083].)

A single arithmetic instruction in the cryptography application causes the processor to multiply a first number by a second number and to implicitly add a high order portion of a previously executed single arithmetic instruction to generate a result that represents the first number multiplied with the second number and summed with the high order portion of a previously executed single arithmetic instruction. (*See, e.g.*, p. 3, paragraph [1010]; FIG. 4A, which illustrates the *umulxc* instruction; and p. 12, paragraphs [1069]-[1070] and Table 1.)

The single arithmetic instruction further causes the processor to store a high order portion of the generated result for use with execution of a subsequent single arithmetic instruction in the cryptography application. (*See, e.g.*, p. 12, Table 1, description of previous carry saved in *exc.*; p. 3, paragraph [1010]; p. 12, paragraphs [1069]-[1070], multiply-chaining for multi-word computations; and p. 8, paragraph [1057], the application of multiply-chaining for accelerating public-key computations.)

Independent claim 61 is directed to a computer-readable storage medium comprising program instructions executable by a processor to implement a cryptography application. (*See, e.g.,* title; p. 1, paragraph [1002]; p. 2, paragraph [1005]; pp. 3-4, paragraph [1011]; p. 8, paragraphs [1057]-[1058]; and p. 16, paragraphs [1082]-[1083].)

A single arithmetic instruction in the cryptography application causes the processor to multiply a first number by a second number and to add implicitly a partial multiplication result from a previously executed single arithmetic instruction and a third number to generate a result that represents the first number multiplied by the second number summed with the partial multiplication result and summed with the third number. (*See, e.g.,* pp. 3-4, paragraph [1011]; FIGs. 4B and 4C, illustrating two different

embodiments of the *umulxck* instruction; p. 13, paragraph [1073] – p. 14, paragraph [1076]; and p. 14, Table 2.)

The single arithmetic instruction further causes the processor to store a high order portion of the generated result for use with execution of a subsequent single arithmetic instruction in the cryptography application. (*See, e.g.*, pp. 3-4, paragraph [1011]; p. 14, Table 2, description of previous carry saved in *exc.*; and p. 15, paragraph [1077], multiply-accumulate-chaining for multi-word multiplication, and its application to public-key computations.)

Independent claim 64 is directed to a processor supporting a cryptography application. (*See, e.g.,* title; p. 1, paragraph [1002]; p. 2, paragraph [1005]; p. 8, paragraphs [1057]-[1058]; and p. 16, paragraphs [1082]-[1083].)

The processor includes means, responsive to a single multiply-accumulate instruction in the cryptography application, for multiplying a first number with a second number and implicitly adding a partial result of a previously executed single multiply-accumulate instruction to generate a result that represents the first number multiplied by the second number summed with the partial result. (*See, e.g.*, p. 3, paragraph [1008]; FIG. 15, multiply-and-accumulate circuit; and p. 29, paragraph [1124].)

The processor also includes means for storing a high order portion of the result for use with execution of a subsequent single multiply-accumulate instruction in the cryptography application. (*See, e.g.*, p. 3, paragraph [1008]; FIG. 15, multiply-and-accumulate circuit; p. 29, paragraph [1124], upper 64-bits of result output to *exc* register 1058; p. 12, paragraphs [1069]-[1070], multiply-chaining for multi-word computations; and p. 8, paragraph [1057], the application of multiply-chaining for accelerating public-key computations.)

Independent claim 65 is directed to a processor supporting a cryptography application. (*See, e.g.,* title; p. 1, paragraph [1002]; p. 2, paragraph [1005]; p. 8, paragraphs [1057]-[1058]; and p. 16, paragraphs [1082]-[1083].)

The processor comprises means, responsive to a single multiply-accumulate instruction in a cryptography application, for multiplying a first number with a second number, for implicitly adding a partial result of a previously executed single multiply-accumulate instruction, and for adding a third number to generate a result that represents the first number multiplied by the second number summed with the partial result and the third number. (*See, e.g.*, p. 3, paragraph [1009]; FIG. 20, multiply-and-accumulate circuit; and pp. 30-31, paragraph [1130].)

The processor also includes means for storing a high order portion of the generated result for use with execution of a subsequent multiply-accumulate instruction in the cryptography application. (*See, e.g.*, p. 3, paragraph [1009]; FIG. 20, multiply-and-accumulate circuit; pp. 30-31, paragraph [1130], upper 64-bits of result output to *exc* register 2005; and p. 15, paragraph [1077], multiply-accumulate-chaining for multi-word multiplication, and its application to public-key computations.)

The summary above describes various examples and embodiments of the claimed subject matter; however, the claims are not necessarily limited to any of these examples and embodiments. The claims should be interpreted based on the wording of the respective claims.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1-65 stand finally rejected under 35 U.S.C. § 101 as being based on non-statutory subject matter.

2. Claims 1-5, 13-23, 30, 33-46, 49-52 and 56-65 stand finally rejected under 35 U.S.C. § 102(e) as being anticipated by Gressel et al. (U.S. Patent 6,748,410) (hereinafter "Gressel").

3. Claims 6-12, 24-29, 31, 32, 47, 48, 53, 54 and 55 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Gressel in view of Stribaek et al. (U.S. Patent 7,181,484) (hereinafter "Stribaek").

VII. ARGUMENT

First ground of rejection:

The Examiner rejected claims 1-65 under 35 U.S.C. § 101 as being based on non-statutory subject matter. Appellants traverse this rejection for at least the following reasons.

Claims 1-42

Claims 1 and 18 each recite a method that is implemented in tangible device. More specifically, claims 1 and 18 recite a method implemented in a device supporting a cryptography application. Thus, the methods of claims 1 and 18 cannot be pure mental processes. Moreover, claims 1 and 18 clearly recite a practical application and a useful, concrete and tangible result. For example, claims 1 and 18 both recite limitations involving the use of a generated result in a cryptography application, which is clearly a practical and useful application. More specifically, claims 1 and 18 both recite storing at least a portion of the generated result, and using the stored at least a portion of the generated result in a subsequent computation in the cryptography application. There can be no doubt that the use of such results in a cryptography application constitutes a practical application.

In *State Street Bank & Trust Co. v. Signature Financial Group, Inc.*, 149 F.3d 1368, 47 USPQ2d 1596 (Fed. Cir. 1998), as discussed in MPEP 2106, the court stated that the relevant claim was statutory because "the transformation of data, representing discrete dollar amounts, by a machine through a series of mathematical calculations into a final share price, constitutes a practical application … because it produces 'a useful, concrete and tangible result' – a final share price". Just like transforming data representing discrete dollar amounts to determine a final share price was considered a practical application and thus statutory in *State Street*, the generation, storage, and use of

a particular result in a cryptography application is a practical application and thus statutory. Claims 1 and 18 clearly recite a practical application in the technological arts.

In addition, rather than merely reciting a mathematical algorithm, claims 1 and 18 are directed to operations that are performed by a device in response to executing a single arithmetic instruction (e.g., a processor instruction supported in the device) that causes specific combinations of mathematical operations to be performed within the device. As described in Appellants' specification, the particular combinations of mathematical operations are commonly used in many different cryptography computations. The use of a single instruction (e.g., supported by circuitry in the device) to cause the execution of these multiple operations may serve to accelerate such applications, therefore producing a useful (specific, substantial, and credible), tangible (not abstract), and concrete (repeatable) result.

As stated in the MPEP Guidelines at IV.C.2.(2).b: "The claim must be examined to see if it includes anything more than a § 101 judicial exception [e.g., abstract idea]. If the claim is directed to a practical application of the § 101 judicial exception producing a result tied to the physical world … then the claim meets the statutory requirement of 35 U.S.C. § 101." (emphasis added). As discussed above, Appellants' claims do not recite mathematical algorithms **in the abstract**, but in a specific, practical application in the art within a device. The methods recited in claims 1 and 18 are clearly directed to a practical application and clearly produce a useful, tangible, and concrete result tied to the physical world.

Claims 43-56

Appellants note that claims 43-56 recite a processor comprising an arithmetic circuit for implementing functionality similar to that recited in claims 1-42, and are therefore directed toward statutory subject matter. In other words, these claims are directed to circuitry responsive to a single arithmetic instruction (i.e., a processor instruction) to execute the operations recited. Appellants assert that such circuitry in a

processor is clearly statutory, since it is directed to structural elements, as well as for the reasons presented above regarding its practical application in the technological arts. Furthermore, for reasons similar as discussed above for claims 1-42, the processors as recited in claims 43-56 clearly produce a useful, concrete and tangible result.

Claims 57-63

Independent claims 57 and 61 recite a storage medium comprising program instructions executable by a processor supporting a cryptography application that cause the processor to implement methods similar to those recited in claims 1-42 in response to executing a single arithmetic instruction included in the program instructions (e.g., a processor instruction, as described above). Also, according to MPEP §2106 IV.B.1(a), a claimed computer-readable medium encoded with a computer program is a computer element which defines structural and functional interrelationships between the computer program and the rest of the computer which permit the computer program's functionality to be realized, and is thus statutory. *See Lowry*, 32 F.3d at 1583-84, 32 USPQ2d at 1035. (*emphasis added*).

Claims 64 and 65

Claims 64 and 65 are directed toward a processor and include means elements, which by statutory definition are structural elements per 35 USC § 112, para 6. Also, for reasons similar to those discussed above, Appellants assert that these claims are directed to statutory subject matter.

For at least the reasons above, Appellants respectfully request the removal of the rejection of claims 1-65 under 35 U.S.C. § 101.

Second ground of rejection:

The Examiner rejected claims 1-5, 13-23, 30, 33-46, 49-52 and 56-65 under 35 U.S.C. § 102(e) as being anticipated by Gressel et al. (U.S. Patent 6,748,410) (hereinafter "Gressel"). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 1, 43, 57, and 64:

1. The Examiner has failed to fully and clearly state his ground of rejection of claim 1 and has therefore failed to establish a *prima facie* case of anticipation given that the burden of proof falls on the Office.

As discussed in Appellants' previous Responses, in the Examiner's remarks regarding the rejection of claim 1 (e.g., on pages 5-6 of the Final Action), he cites a large number of passages in Gressel as teaching: "feedback of a previous operation into next operation", "arithmetic operation or instructions", "arithmetic structure", "multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication", "register usage", "XOR operations" "redundant representation of numbers", "acceleration, improvements of arithmetic operations", "arithmetic operations utilized to generate cryptographic key(s)", and "processor utilization for key generation", without describing which of these passages (or elements described therein) he believes discloses each of the limitations of claim 1 or how he interprets these passages to teach the specific limitations of claim 1 **as arranged in the claim.** The statute clearly places the burden of proof on the Patent Office to prove a *prima facie* rejection. *In re Warner*, 154 USPQ 173, 177 (C.C.P.A. 1967), *cert. denied*, 389 U.S. 1057 (1968). The Examiner's vague assertions which lack a clear mapping between the teachings of Gressel and Appellants' claims cannot be said to establish a *prima facie* case of anticipation.

2. The cited art clearly fails to disclose the limitations of claim 1.

Appellants assert that many of the Examiner's citations do not teach the features suggested by the Examiner. For example, the Examiner cites column 53, lines 13-19 and 49-51 as teaching "feedback of a previous operation into next operation." These passages describe the operation of a linear feedback shift register. **They have absolutely nothing to do with the claimed limitation, *adding implicitly a partial result from a previously executed single arithmetic instruction to generate a result that represents the first number multiplied by the second number summed with the partial result***, much less performing such adding in response to executing the single instruction of claim 1 (i.e., the same single instruction that causes the multiplication operation also recited in claim 1).

Similarly, the Examiner cites column 2, lines 31-37 as teaching "multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication." **This passage does not teach anything about high order bits, low order bits, or utilizing partial results from previous multiplication, as the Examiner has suggested.** Instead, this passage states, in its entirety:

> Further in accordance with a preferred embodiment of the present invention, the employing step includes multiplying a first integer of any bit length by a second integer of any bit length to obtain a first product, multiplying a third integer of any bit length by a fourth integer of any bit length to obtain a second product, and summing the first and second products with a fifth integer of any bit length to obtain a sum.

In other words, this passage describes that pairs of integers of any length may be multiplied together, and then the results may be added together with another integer (the fifth integer) of any bit length. There is nothing in this passage about any of the integers being partial results of a previous instruction, as the Examiner suggests. **Therefore, this passage teaches nothing about implicitly adding a partial result of a previous instruction**, much less doing so in response to executing the (same) single instruction of claim 1.

In yet another example, the Examiner cites column 29, lines 43-49 as teaching "redundant representation of numbers." **This passage actually describes the use of a redundant register. It has absolutely nothing to do with redundant representation of numbers, as the Examiner suggests.** This also has nothing to do with the limitations of claim 1.

3. The descriptions of individual features listed by the Examiner do not teach the specific combination of limitations recited in claim 1, as arranged in the claim.

The Examiner is clearly attempting a piecemeal reconstruction of Appellants' invention in hindsight without consider the claimed invention as a whole. Such reconstruction is improper. *See, e.g., Interconnect Planning Corp. v. Feil*, 774 F.2d 1132, 1143, 227 USPQ 543, 551 (Fed. Cir. 1985). For example, a reference in column 3, lines 1-7 to "generate a sum and to feed in the sum to an $(i+1)^{th}$ Montgomery multiplication operation" does not teach the specific limitations regarding adding implicitly a partial result from a previously executed single arithmetic instruction in response to executing the (same) single arithmetic instruction that causes the recited multiplication operation, much less adding a partial result that comprises a high order portion of a result of the previous instruction.

In another example, the Examiner's citations that refer generally to various "arithmetic operation or instructions" or "arithmetic structure" clearly do not teach any of the specific limitations of Appellants' claim.

In the Advisory Action, the Examiner states:

> In any event, the Gressel prior art discloses the acceleration of arithmetic operations by the prior art invention. (see Gressel col. 1, lines 39-45; col. 5, lines 23-25: arithmetic operation acceleration) In addition, the Gressel prior art discloses that the arithmetic operations can be utilized for the generation of cryptographic keys as per the specification and the processing of a cryptographic application as per claim limitations. (see

> Gressel col. 3, lines 24-35: cryptographic application (cryptographic key generation, accelerated)).

Appellants assert, however, that this is irrelevant, since, as discussed in detail herein, Gressel does not teach that this acceleration is done in the manner recited in Appellants' claims. Gressel's methods for acceleration of cryptographic key generation and arithmetic operations do not teach the specific limitations recited in Appellants claims, e.g., those involving the recited single arithmetic instruction, execution of which results in performance of the specific combination of arithmetic operations recited in the claims.

> In the Advisory Action, the Examiner further submits:
>
> Applicant's claim limitations disclose the completion of arithmetic operations (addition, subtraction, multiplication) with regular and extended register. Each type of arithmetic operation disclosed by applicants claim invention is disclosed by the Gressel and Stribaek prior art references. The operations of addition, subtraction, and multiplication, XOR operations are arithmetic operations well known in the art. The performance of these arithmetic operations is the basis for the functions performed by computer systems and Information Technology systems. There is nothing novel or unique about the completion of these operations. In the processing of arithmetic operations, the reuse of the results (full word, partial word) of a first arithmetic operation as input to a second arithmetic operation is well known in the art. There is nothing novel or unique about the reuse of a result (full word, partial word) from one operation as input to a second operation. The operation of multiple instructions within a single combined instruction is well known in the art. There is nothing novel or unique about this particular structure for an arithmetic instruction. The arithmetic instruction, XOR, is a standard operation performed by a computer system. There is nothing novel or unique about this particular arithmetic operation. All of the arithmetic operations disclosed by applicant's invention are well known in the art. There is nothing novel or unique.

However, Appellants' claims do not recite mere "operations of addition, subtraction, and multiplication, XOR operations are arithmetic operations." A prior art machine that performs such operations does not function in the manner recited in Appellants' claim 1. The Examiner is mischaracterizing the invention, as claimed. Appellants' claims are directed to methods and systems for employing a single arithmetic instruction (e.g., a

umulxc instruction, or a *umulxck* instruction) in a cryptography application to perform the specific combination of arithmetic operations recited. This is neither taught nor inherently present in the prior art. The Examiner's remarks appear to be directed to a method for performing these arithmetic operations by executing a series of individual known arithmetic instructions, which is in direct contrast to Appellants' claimed invention.

The Examiner's remarks regarding VLIW instructions are similarly misplaced. The combination of arithmetic operations recited in Appellants' claims could not be executed as arranged in the claims using a VLIW instruction, as the operations executed as a result of a single instruction in a VLIW architecture are by definition independent of each other. This is clearly not the case with the arithmetic operations recited in Appellants' claims, in which **results** of one operation specified by the instruction (a multiply operation) are added to one or more other operands specified in the same single instruction in response to executing the single instruction. Appellants assert that one of ordinary skill in the art would easily recognize that Appellants' claimed invention could not be carried out by merely combining the recited arithmetic operations into a single VLIW-type instruction. **Furthermore, it is improper for the Examiner to rely on the teachings of a VLIW instruction or architecture, which are not disclosed in the Gressel reference, when rejecting the claims under 35 U.S.C. § 102(e) as being anticipated by Gressel. Appellants assert that Gressel clearly fails to disclose the actions performed in response to a single arithmetic instruction as recited in Appellants' claim 1.**

Anticipation requires the presence in a single prior art reference disclosure of each and every limitation of the claimed invention, **arranged as in the claim**. M.P.E.P 2131; *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The **identical invention must** be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). As discussed above, Gressel clearly does not disclose all of the specific limitations of claim 1 as arranged therein.

For at least the reasons above, Gressel cannot be said to anticipate claim 1 and removal of the rejection there is respectfully requested.

Claims 43, 57, and 64 include limitations similar to those recited in claim 1 and discussed above, and were rejected for the same reasons as claim 1. Therefore, the arguments presented above apply with equal force to these claims, as well.

Claims 18, 50, 61, and 65:

1. The cited art clearly fails to disclose all the limitations of these claims, as arranged in the claims.

Claim 18 includes limitations similar to those recited in claim 1 and discussed above, and was rejected for the same reasons as claim 1. Therefore, Appellants traverse this rejection for at least the reasons presented above regarding limitations in this claims that are similar to those in claim 1. In addition, claim 18 recites *adding a third number to generate a result that represents the first number multiplied by the second number summed with the partial result and the third number.* **The Examiner does not include any additional remarks regarding this limitation or any additional citations in the reference to teach it.** Therefore, the Examiner has failed to state a *prima facie* rejection of claim 18. Appellants assert that the Examiner's citations regarding feedback from previous operations and the multiplication and/or addition of integers of any bit length teach nothing about a single arithmetic instruction that results in the operations recited in claim 18.

For at least the reasons above, Gressel cannot be said to anticipate claim 18 and removal of the rejection thereof is respectfully requested.

Claims 50, 61, and 65 include limitations similar to those recited in claim 18 and discussed above, and were rejected for the same reasons as claim 18. Therefore, the arguments presented above apply with equal force to these claims, as well.

Claims 2 and 19:

1. The cited art clearly fails to disclose *performing the adding of the partial result as part of addition operations performed for the multiplying of the first and second number*, as recited in claim 2.

The Examiner again cites column 2, lines 31-37 as teaching this limitation. **However, as discussed above, this passage teaches nothing about adding a partial result of an operation at all, much less adding the partial result as part of addition operations performed for multiplying the first and second number, as recited in claim 2.**

For at least the reasons above, Gressel cannot be said to anticipate claim 2 and removal of the rejection thereof is respectfully requested.

Claim 19 includes limitations similar to those recited in claim 2 and discussed above, and was rejected for the same reasons as claim 2. Therefore, the arguments presented above apply with equal force to this claim, as well.

Claims 3, 20, 45, and 52:

1. The cited art clearly fails to disclose *wherein the partial result is in redundant number representation.*

The Examiner again cites column 29, lines 43-49 as teaching this limitation. **However, as discussed above, this passage describes a redundant register. It has**

absolutely nothing to do with a partial result being in redundant number representation.

For at least the reasons above, Gressel cannot be said to anticipate claim 3 and removal of the rejection there is respectfully requested.

Claims 20, 45, and 52 include limitations similar to those recited in claim 3. Therefore, the arguments presented above apply with equal force to these claims, as well.

Claim 4:

1. **The cited art clearly fails to disclose *wherein said adding the partial result comprises adding the partial result to a multiplication result of the first and second numbers.***

The Examiner again cites column 2, lines 31-37 as teaching this limitation. **However, as discussed above, this passage teaches nothing about adding a partial result of an operation at all, much less adding the partial result to a multiplication result of the first and second numbers, as recited in claim 4.**

For at least the reasons above, Gressel cannot be said to anticipate claim 4 and removal of the rejection thereof is respectfully requested.

Claims 5, 23, and 51:

1. **The cited art clearly fails to disclose *wherein said storing at least a portion of the generated result comprises storing a high order portion of the generated result as a next partial result for use with execution of a subsequent single arithmetic instruction.***

The Examiner again cites column 3, lines 1-7, and column 53, lines 13-19 and 49-51, "feedback of a previous operation into a next operation". **However, as discussed above, these passage teach nothing about storing a portion of a result for use in a subsequent instruction, much less the specific limitations recited in claim 5.**

For at least the reasons above, Gressel cannot be said to anticipate claim 5 and removal of the rejection thereof is respectfully requested.

Claims 23 and 51 include limitations similar to those recited in claim 5. Therefore, the arguments presented above apply with equal force to these claims, as well.

Claim 13:

1. **The cited art clearly fails to disclose *the single arithmetic instruction is a single multiply-accumulate instruction; wherein the first and second numbers are specified in the single multiply-accumulate instruction as first and second source registers, and a low order portion of the result is stored in a destination location specified in the single multiply-accumulate instruction.***

The Examiner again cites column 3, lines 1-7, and column 53, lines 13-19 and 49-51, "feedback of a previous operation into a next operation", along with col. 3, lines 28-32, and column 11, lines 7-11 and 40-49 "arithmetic operation or instructions" as teaching these limitations. **However, as discussed above, the Examiner's citations regarding feedback do not teach the use of a partial result in a subsequent operation, as in claim 13. In addition, the general references to arithmetic operations or instructions in columns 3 and 11 teach nothing about the specific limitations recited in claim 13 regarding a single arithmetic instruction.**

For at least the reasons above, Gressel cannot be said to anticipate claim 13 and removal of the rejection thereof is respectfully requested.

Claim 14:

1. **The cited art clearly fails to disclose *wherein the first and second numbers are n-bit numbers, n being a positive integer, and wherein the high order portion of the generated result is an n-bit portion.***

The Examiner again cites column 2, lines 31-37. **However, as discussed above, this passage teaches nothing about a partial result comprising a high order portion of a generated result, much less one having a same number of bits as the recited first and second numbers.**

For at least the reasons above, Gressel cannot be said to anticipate claim 14 and removal of the rejection thereof is respectfully requested.

Claims 15 and 59:

1. **The cited art clearly fails to disclose *in response to executing the subsequent single arithmetic instruction, multiplying third and fourth numbers specified by the subsequent single arithmetic instruction and adding implicitly the next partial result to generate a second result that represents the third number multiplied by the fourth number summed with the next partial result.***

The Examiner again cites column 2, lines 31-37, along with col. 3, lines 28-32, and column 11, lines 7-11 and 40-49 "arithmetic operation or instructions" as teaching these limitations. **However, these citations teach nothing about a subsequent single arithmetic instruction (i.e., one in which a partial result of the earlier recited single arithmetic instruction is used) much less that the specific operations recited in claim 15 are performed in response to executing such an instruction.**

For at least the reasons above, Gressel cannot be said to anticipate claim 15 and removal of the rejection thereof is respectfully requested.

Claim 59 includes limitations similar to those recited in claim 15. Therefore, the arguments presented above apply with equal force to this claim, as well.

Claims 16 and 38:

1. The cited art clearly fails to disclose *storing the high order portion of the second result to be implicitly added in response to executing another subsequent single arithmetic instruction.*

The Examiner again cites column 3, lines 1-7, and column 53, lines 13-19 and 49-51, "feedback of a previous operation into a next operation", along with column 2, lines 31-37. However, these passages teach nothing about a second result of a subsequent single arithmetic instruction, and therefore, nothing about storing the high order portion of such a second result to be added to yet another subsequent single arithmetic instruction, as recited in claim 16.

For at least the reasons above, Gressel cannot be said to anticipate claim 16 and removal of the rejection thereof is respectfully requested.

Claim 38 includes limitations similar to those recited in claim 16. Therefore, the arguments presented above apply with equal force to this claim, as well.

Claims 17 and 42:

1. The cited art clearly fails to disclose *wherein the multiplying and adding are implemented to support XOR operations for binary polynomial fields.*

The Examiner cites column 8, lines 59-60 and column 53, lines 13-19, "XOR operations" as teaching this limitation. These passages describe XOR circuitry in the system of Gressel that is used during a multiplication operation and in a Johnson counter.

They have absolutely nothing to do with XOR operations (or any other operations) for binary polynomial fields, or the multiplying and adding of Appellants' claims being implemented to support such operations.

For at least the reasons above, Gressel cannot be said to anticipate claim 17 and removal of the rejection thereof is respectfully requested.

Claim 42 includes limitations similar to those recited in claim 17. Therefore, the arguments presented above apply with equal force to this claim, as well.

Claim 21:

1. The cited art clearly fails to disclose *performing the adding of the third number as part of the addition performed for the multiplying of the first and second number.*

The Examiner again cites column 2, lines 31-37 as teaching "multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication." **However, as discussed above, the general references in this passage do not teach or suggest anything about the specific limitations of claim 21 regarding the adding of the third number.**

For at least the reasons above, Gressel cannot be said to anticipate claim 21 and removal of the rejection thereof is respectfully requested.

Claim 22:

1. The cited art clearly fails to disclose *wherein said adding the partial result comprises adding the partial result after generation of a multiplication result of multiplying the first and second numbers.*

The Examiner again cites column 3, lines 1-7, and column 53, lines 13-19 and 49-51, "feedback of a previous operation into a next operation" and column 2, lines 31-37 as teaching "multiplication two values, summing two values utilizing partial (i.e. bit operations, any bit length, high order bits, low order bits) results from previous multiplication." **However, as discussed above, the general references in these passages do not teach or suggest anything about the specific limitations of claim 22.**

For at least the reasons above, Gressel cannot be said to anticipate claim 22 and removal of the rejection thereof is respectfully requested.

Claims 30, 33, 34, 35, 39, 40, 41, 49, 56, and 58:

1. The cited art clearly fails to disclose the limitations of claims 30, 33, 34, 35, 39, 40, 41, 49, 56, and 58.

These claims recite limitations involving the implicit and explicit identification by the single arithmetic instruction and in subsequent single arithmetic instructions of the various operands that are multiplied and added by a device or processor in response to executing these instructions, as in Appellants' claimed invention. **Appellants assert that none of the Examiner's citations, or anything else in Gressel, discloses the specific limitations recited in these claims regarding the identification of these operands in single arithmetic instructions.**

Claims 36, 60, 62, and 63:

1. The cited art clearly fails to disclose *in response to executing the subsequent single arithmetic instruction, multiplying a fourth number and a fifth number, the fourth number being specified by the subsequent single arithmetic instruction, adding implicitly the next partial multiplication result, and adding a sixth number to generate a second result, the second result representing the fourth number*

multiplied by the fifth number summed with the next partial result and the sixth number.

The Examiner again cites col. 3, lines 28-32, and column 11, lines 7-11 and 40-49 "arithmetic operation or instructions" as teaching these limitations. **Again, Appellants assert that the generic references to arithmetic operations or instructions in these passages teach absolutely nothing about the specific limitations recited in claim 36 regarding the subsequent single arithmetic instruction.**

For at least the reasons above, Gressel cannot be said to anticipate claim 36 and removal of the rejection thereof is respectfully requested.

Claims 60, 62, and 63 include additional limitations reciting additional multiplication and implicit addition operations that are performed in response to executing a subsequent single arithmetic instruction. **For reasons similar to those recited above regarding claim 36, Appellants assert that nothing in the Examiner's citations, or elsewhere in Gressel, discloses these subsequent single arithmetic operations and corresponding multiplication and implicit addition operations.**

For at least the reasons above, Gressel cannot be said to anticipate claims 60, 62, and 63 and removal of the rejection thereof is respectfully requested.

Claim 37:

1. The cited art clearly fails to disclose *wherein the fifth number and the second number are equal.*

The Examiner again cites column 29, lines 43-49 "representation of numbers". However, as discussed above, this passage has nothing to do with "representation of numbers." Instead, it describes "four main serial main registers," one of which is redundant. In addition, claim 37 has nothing to do with "representation of numbers", as

suggested by the Examiner. **This passage has nothing to do with the limitations of claim 37.**

For at least the reasons above, Gressel cannot be said to anticipate claim 37 and removal of the rejection thereof is respectfully requested.

Claims 44 and 46:

1. The cited art clearly fails to disclose the limitations of claims 44 and 46.

Claims 44 and 46 include limitations regarding an extended carry register, in which at least a portion of the generated result is stored. However, in remarks regarding claim 6, the Examiner admits that Gressel does not specifically disclose an extended carry register and relies on Stribaek to teach this feature in column 5, lines 41-45. **Therefore, the rejection of claims 44 and 46 under 35 U.S.C. § 102(e) as being anticipated by Gressel is improper.**

For at least the reasons above, Gressel cannot be said to anticipate claims 44 and 46 and removal of the rejection thereof is respectfully requested.

Third ground of rejection:

The Examiner rejected claims 6-12, 24-29, 31, 32, 47, 48, 53, 54 and 55 under 35 U.S.C. § 103(a) as being unpatentable over Gressel in view of Stribaek et al. (U.S. Patent 7,181,484) (hereinafter "Stribaek"). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 6 and 53:

1. **The cited art clearly fails to teach or suggest *wherein said storing the high order portion of the generated result comprises storing the high order portion of the generated result into an extended carry register for use with execution of the subsequent single arithmetic instruction***, as recited in claim 6.

The Examiner submits that Gressel teaches storing the high order portion of the generated result for use with execution of the subsequent single arithmetic instruction. **However, as discussed above, Gressel does not teach this limitation.**

The Examiner admits that Gressel does not specifically disclose an extended carry register and relies on Stribaek to teach this feature in column 5, lines 41-45. The Examiner submits that it would have been obvious to one of ordinary skill in the art to modify Gressel as taught by Stribaek to enable the capability for the usage of an extended carry register to enable the capability for extended precision arithmetic calculations due to extensive and increasing usage of public key cryptography. While Stribaek does describe an extended carry register and its usefulness in extended precision arithmetic, it does not teach or suggest the specific use of such a register recited in claim 6. In fact, the Examiner citation states, "Similarly, the instruction format for MTLHX ("Move to Lo, Hi and Extended Carry") is shown in FIG. 10B. When executed, an appropriate number of bits (e.g., eight) of the value in HI register 2022 are written into the ACX register 2021." This describes that a specific move-type instruction is used for loading the extended carry register. **This clearly does not teach that a partial result is (or could be) stored in an extended carry register in response to executing the single arithmetic instruction that also performs the addition and multiplication operations recited in claim 1.** Therefore, adding this extended carry register and its special load operation to the system of Gressel clearly would not result in the claimed invention.

To establish a *prima facie* obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180

U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. Appellants assert that, as discussed above, the cited references do not teach the limitations of claim 6, whether taken alone or in combination.

For at least the reasons above, the rejection of claim 6 is unsupported by the cited art and removal thereof is respectfully requested.

Claim 53 includes limitations similar to those recited in claim 6. Therefore, the arguments presented above apply with equal force to this claim, as well.

Claims 7, 8, 10, 12, 24-27, 29, 44, and 54:

1. The cited art clearly fails to teach or suggest all the limitations of claims 7, 8, 10, 12, 24-27, 29, 44, and 54.

Claims 7, 8, 10, 12, 24-27, 29, 44, and 54 recite additional limitations involving an extended carry register into which partial results are loaded and/or from which they are retrieved in response to executing a single arithmetic instruction. Therefore, the arguments presented above regarding claim 6 apply with equal force to these claims as well.

Claim 9:

1. The cited art clearly fails to teach or suggest *wherein the third and fourth numbers are zero*.

The Examiner again cites Gressel column 29, lines 43-49 "representation of numbers". However, as discussed above, this passage has nothing to do with "representation of numbers." Instead, it describes "four main serial main registers," one of which is redundant. In addition, claim 9 has nothing to do with "representation of

numbers", as suggested by the Examiner. **This passage has nothing to do with the limitations of claim 9.**

For at least the reasons above, the rejection of claim 9 is unsupported by the cited art and removal thereof is respectfully requested.

Claims 11 and 28:

1. The cited art clearly fails to teach or suggest *selecting one of a plurality of extended carry registers as the extended carry register.*

The Examiner again cites Stribaek column 5, lines 41-45, "extended carry operations" as teaching the extended carry register of claim 11. However, Stribaek does not teach or suggest a plurality of extended carry registers, but describes only one extended carry register, used in a different context than that of Appellants' claimed invention. In addition, nothing in Stribaek or Gressel teaches of suggest selecting one of such a plurality of extended carry registers, as recited in claim 11, much less selecting one of a plurality of extended carry registers to be used in the context recited in Appellants' claims.

For at least the reasons above, the rejection of claim 11 is unsupported by the cited art and removal thereof is respectfully requested.

Claim 28 includes limitations similar to those recited in claim 11. Therefore, the arguments presented above apply with equal force to this claim, as well.

Claims 47 and 55:

1. The cited art clearly fails to teach or suggest *wherein the extended carry register has an associated dirty bit indicating whether contents of the extended carry register need to be saved on a context switch.*

The Examiner submits that Gressel discloses the register has an associated dirty bit indicating whether contents of a register need to be saved on a context switch in column 24, lines 4-10, as "switch based on context of data." This passage states, in its entirety:

> A primary step in masking squares and multiplication is to execute a squaring operation in a mode wherein all rotating registers and computational devices are exercised in exactly the same manner for squaring and multiplying, the only difference being the settings of data switches which choose relevant data for computation and not using [trashing] the irrelevant data.

It is clear that this passage has absolutely nothing to do with the limitations recited in claim 47 regarding a dirty bit in a register, or such a bit indicating whether the contents of the register need to be saved on a context switch. In addition, as discussed above regarding claim 6, the cited references fail to teach or suggest an extended carry register used in the manner recited in Appellants' claims.

For at least the reasons above, the rejection of claim 47 is unsupported by the cited art and removal thereof is respectfully requested.

Claim 55 includes limitations similar to those recited in claim 47. Therefore, the arguments presented above apply with equal force to this claim, as well.

Claims 31-32 and 48:

Appellants traverse the rejection of these claims for at least the reasons presented above regarding the claims from which they depend.

CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-65 was erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/6000-32301/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Appellants

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8850

Date: February 20, 2008

VIII. CLAIMS APPENDIX

The claims on appeal are as follows.

1. A method implemented in a device supporting a cryptography application, the method comprising:

in response to executing a single arithmetic instruction,

multiplying a first number by a second number; and

adding implicitly a partial result from a previously executed single arithmetic instruction to generate a result that represents the first number multiplied by the second number summed with the partial result, wherein the partial result comprises a high order portion of a result of the previously executed single arithmetic instruction;

storing at least a portion of the generated result; and

using the stored at least a portion of the generated result in a subsequent computation in the cryptography application.

2. The method as recited in claim 1 further comprising performing the adding of the partial result as part of addition operations performed for the multiplying of the first and second number.

3. The method as recited in claim 1 wherein the partial result is in redundant number representation.

4. The method as recited in claim 1, wherein said adding the partial result comprises adding the partial result to a multiplication result of the first and second numbers.

5. The method as recited in claim 1, wherein said storing at least a portion of the generated result comprises storing a high order portion of the generated result as a next partial result for use with execution of a subsequent single arithmetic instruction.

6. The method as recited in claim 5, wherein said storing the high order portion of the generated result comprises storing the high order portion of the generated result into an extended carry register for use with execution of the subsequent single arithmetic instruction.

7. The method as recited in claim 6, further comprising retrieving an indication of a current value of the extended carry register by executing another single arithmetic instruction that multiplies a third number by a fourth number and that implicitly adds current contents of the extended carry register to generate a second result that represents the third number multiplied by the fourth number summed with the current contents of the extended carry register.

8. The method as recited in claim 7, wherein a low order portion of the second result contains the indication of the current value of the extended carry register.

9. The method as recited in claim 7, wherein the third and fourth numbers are zero.

10. The method as recited in claim 6, further comprising loading the extended carry register with a predetermined value by executing another single arithmetic instruction that multiplies a third number by a fourth number, and that implicitly adds a current value of the extended carry register, to generate a result that represents the third number multiplied by the fourth number summed with the current value of the extended

carry register and to store it in the extended carry register, thereby loading the extended carry register with the predetermined value.

11. The method as recited in claim 6, further comprising selecting one of a plurality of extended carry registers as the extended carry register.

12. The method as recited in claim 6, further comprising accessing the extended carry register via at least one of a load instruction and a store instruction.

13. The method as recited in claim 1, wherein the single arithmetic instruction is a single multiply-accumulate instruction; wherein the first and second numbers are specified in the single multiply-accumulate instruction as first and second source registers, and a low order portion of the result is stored in a destination location specified in the single multiply-accumulate instruction.

14. The method as recited in claim 5, wherein the first and second numbers are n-bit numbers, n being a positive integer, and wherein the high order portion of the generated result is an n-bit portion.

15. The method as recited in claim 5 further comprising:

in response to executing the subsequent single arithmetic instruction,

multiplying third and fourth numbers specified by the subsequent single arithmetic instruction and adding implicitly the next partial result to generate a second result that represents the third number multiplied by the fourth number summed with the next partial result.

16. The method as recited in claim 15, further comprising storing the high order portion of the second result to be implicitly added in response to executing another subsequent single arithmetic instruction.

17. The method as recited in claim 1 wherein the multiplying and adding are implemented to support XOR operations for binary polynomial fields.

18. A method implemented in a device supporting a cryptography application, the method comprising:

in response to executing a single arithmetic instruction,

multiplying a first number by a second number;

adding implicitly a partial result from a previously executed single arithmetic instruction, wherein the partial result comprises a high order portion of a result of the previously executed single arithmetic instruction;

adding a third number to generate a result that represents the first number multiplied by the second number summed with the partial result and the third number;

storing at least a portion of the generated result; and

using the stored at least a portion of the generated result in a subsequent computation in the cryptography application.

19. The method as recited in claim 18 further comprising performing the adding of the partial result as part of addition performed for the multiplying of the first and second number.

20. The method as recited in claim 18 wherein the partial result is stored in a redundant number representation.

21. The method as recited in claim 18 further comprising performing the adding of the third number as part of the addition performed for the multiplying of the first and second number.

22. The method as recited in claim 18, wherein said adding the partial result comprises adding the partial result after generation of a multiplication result of multiplying the first and second numbers.

23. The method as recited in claim 18, wherein said storing at least a portion of the generated result comprises storing a high order portion of the generated result as a next partial multiplication result for use with execution of a subsequent single arithmetic instruction.

24. The method as recited in claim 23 wherein said storing the high order portion of the generated result comprises storing the high order portion of the generated result into an extended carry register for use with execution of the subsequent arithmetic instruction.

25. The method as recited in claim 24, further comprising retrieving an indication of a current value of the extended carry register by executing another single arithmetic instruction that multiplies a fourth number by a fifth number, that implicitly adds current contents of the extended carry register, and that adds a sixth number to generate a second result that represents the fourth number multiplied by the fifth number summed with the current contents of the extended carry register and the sixth number.

26. The method as recited in claim 25, wherein a low order portion of the second result contains the indication of the current value of the extended carry register.

27. The method as recited in claim 24, further comprising loading the extended carry register with a predetermined value by executing another single arithmetic

instruction that multiplies a fourth number by a fifth number, that implicitly adds a current value of the extended carry register, and that adds a sixth number, to generate a second result that represents the third number multiplied by the fourth number summed with the current value of the extended-carry register and summed with the sixth number and to store it in the extended carry register, thereby loading the extended carry register with the predetermined value.

28. The method as recited in claim 24, further comprising selecting one of a plurality of extended carry registers as the extended carry register.

29. The method as recited in claim 24, further comprising accessing the extended carry register via at least one of a load instruction and a store instruction.

30. The method as recited in claim 24 wherein the second number is implicitly identified in the single arithmetic instruction.

31. The method as recited in claim 30, further comprising accessing a special register storing the second number via at least one of a load instruction and a store instruction.

32. The method as recited in claim 18, further comprising accessing a special register storing the second number via at least one of a load instruction and a store instruction.

33. The method as recited in claim 18, wherein the first and third numbers are specified in the single arithmetic instruction as first and second source registers and a low order portion of the generated result is stored in a destination location specified in the single arithmetic instruction.

34. The method as recited in claim 33 wherein the partial result from a previously executed single arithmetic instruction is implicitly specified by the single

arithmetic instruction and wherein the second number is explicitly specified by a third source register in the single arithmetic instruction.

35. The method as recited in claim 33 wherein the partial result from a previously executed single arithmetic instruction is implicitly specified by the single arithmetic instruction and the second number is implicitly specified by the single arithmetic instruction.

36. The method as recited in claim 23 further comprising:

in response to executing the subsequent single arithmetic instruction,

multiplying a fourth number and a fifth number, the fourth number being specified by the subsequent single arithmetic instruction, adding implicitly the next partial multiplication result, and adding a sixth number to generate a second result, the second result representing the fourth number multiplied by the fifth number summed with the next partial result and the sixth number.

37. The method as recited in claim 36 wherein the fifth number and the second number are equal.

38. The method as recited in claim 36, further comprising storing a high order portion of the second result to be implicitly added in response to executing another subsequent single arithmetic instruction.

39. The method as recited in claim 18, wherein the first number is specified in the single arithmetic instruction in a first source register, the second number is contained in a special register and is not specified in the single arithmetic instruction, the third number is specified as a second source register in the single arithmetic instruction, and a

low order portion of the generated result is stored in a destination location specified in the single arithmetic instruction.

40. The method as recited in claim 18 wherein the first, second, and third numbers are specified by source operands in the single arithmetic instruction.

41. The method as recited in claim 18, wherein a destination location and one of the first number, the second number, and the third number are specified by one operand in the single arithmetic instruction.

42. The method as recited in claim 18 wherein the multiplying and adding operations are implemented for binary polynomial fields.

43. A processor, comprising an arithmetic circuit, the processor configured to be responsive to execution of a single arithmetic instruction to:

cause the arithmetic circuit to multiply a first number and a second number and to add implicitly a high order portion of a partial result from a previously executed single arithmetic instruction, thereby generating a result that represents the first number multiplied by the second number summed with the high order portion of the partial result;

store at least a portion of the generated result; and

use the stored at least a portion of the generated result in a subsequent computation.

44. The processor as recited in claim 43, wherein to store at least a portion of the generated result, the processor is further responsive to the single arithmetic instruction to store a high order portion of the generated result into an extended carry register for use with execution of a subsequent single arithmetic instruction.

45. The processor as recited in claim 43 wherein the high order portion of the previously executed single arithmetic instruction is stored in a redundant number representation.

46. The processor as recited in claim 45, wherein the extended carry register is a register accessible via a processor instruction.

47. The processor as recited in claim 45, wherein the extended carry register has an associated dirty bit indicating whether contents of the extended carry register need to be saved on a context switch.

48. The processor as recited in claim 43, wherein the extended carry register is a special register.

49. The processor as recited in claim 43 wherein the first and second numbers are specified in the single arithmetic instruction as first and second source registers.

50. A processor, comprising an arithmetic circuit, the processor configured to be responsive to execution of a single arithmetic instruction to:

cause the arithmetic circuit to multiply a first number and a second number, to add a third number, and to implicitly add a high order portion of a previous result from a previously executed single arithmetic instruction, thereby generating a result that represents the first number multiplied with the second number, summed with the high order portion of the previous result and with the third number;

store at least a portion of the generated result; and

use the stored at least a portion of the generated result in a subsequent computation.

51. The processor as recited in claim 50, wherein to store at least a portion of the generated result, the processor is configured to store a high order portion of the generated result for use with execution of a subsequent single arithmetic instruction.

52. The processor as recited in claim 50 wherein the high order portion of the previous result is stored in a redundant number representation.

53. The processor as recited in claim 50, wherein the processor is configured to store the high order portion of the generated result into an extended carry register.

54. The processor as recited in claim 50, wherein the extended carry register is a special register accessible by the processor via at least one of: load instructions and store instructions.

55. The processor as recited in claim 50, wherein the extended carry register has an associated dirty bit indicating whether contents of the extended carry register need to be saved on a context switch.

56. The processor as recited in claim 50, wherein the first number is specified in the single arithmetic instruction as a first source register, the second number is contained in a logically local register and is not specified in the single arithmetic instruction, the third number is specified as a second source register in the single arithmetic instruction, and a low order portion of the result is stored in a destination location specified in the single arithmetic instruction.

57. A computer-readable storage medium, comprising program instructions executable by a processor to implement a cryptography application:

wherein a single arithmetic instruction in the cryptography application causes the processor to multiply a first number by a second number and to implicitly add a high order portion of a previously executed single arithmetic instruction to generate a result that represents the first number multiplied with the second number and summed with the high order portion of a previously executed single arithmetic instruction,

wherein the single arithmetic instruction further causes the processor to store a high order portion of the generated result for use with execution of a subsequent single arithmetic instruction in the cryptography application.

58. The storage medium as recited in claim 57, wherein the single arithmetic instruction includes a first source operand and a second source operand, specifying the first number and the second number, and a destination operand wherein the single arithmetic instruction further causes the processor to store a low order portion of the generated result in a location specified by the destination operand.

59. The storage medium as recited in claim 57, wherein the subsequent single arithmetic instruction causes the processor executing the subsequent single arithmetic instruction to multiply a third number by a fourth number and implicitly add the high order portion of the result.

60. The storage medium as recited in claim 59; wherein another single arithmetic instruction in the cryptography application causes the processor to multiply a fifth number by a sixth number and to generate another result without implicitly adding another high order portion of another previously executed result, and to store a high order portion of the other result for use with another subsequent single arithmetic instruction.

61. A computer-readable storage medium, comprising program instructions executable by a processor to implement a cryptography application:

wherein a single arithmetic instruction in the cryptography application causes the processor to:

multiply a first number by a second number;

add implicitly a partial multiplication result from a previously executed single arithmetic instruction and a third number to generate a result that represents the first number multiplied by the second number summed with the partial multiplication result and summed with the third number; and

store a high order portion of the generated result for use with execution of a subsequent single arithmetic instruction in the cryptography application.

62. The storage medium as recited in claim 61, wherein the subsequent second single arithmetic instruction causes the processor to multiply a fourth number by the second number, to add a fifth number, and to implicitly add the high order portion of the generated result.

63. The storage medium as recited in claim 61, wherein the subsequent single arithmetic instruction causes the processor to multiply a fourth number by a fifth number, to add a sixth number, and to implicitly add the high order portion of the generated result.

64. A processor supporting a cryptography application, comprising:

means, responsive to a single multiply-accumulate instruction in the cryptography application, for multiplying a first number with a second number and implicitly adding a partial result of a previously executed single multiply-accumulate instruction to generate a .result that represents the first number multiplied by the second number summed with the partial result; and

means for storing a high order portion of the result for use with execution of a subsequent single multiply-accumulate instruction in the cryptography application.

65. A processor supporting a cryptography application, comprising:

means, responsive to a single multiply-accumulate instruction in a cryptography application, for multiplying a first number with a second number, for implicitly adding a partial result of a previously executed single multiply-accumulate instruction, and for adding a third number to generate a result that represents the first number multiplied by the second number summed with the partial result and the third number; and

means for storing a high order portion of the generated result for use with execution of a subsequent multiply-accumulate instruction in the cryptography application.

IX. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

X. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.